# Lab: API and Integration Tests

This document defines the homework assignments from the ["QA Automation" Course @ Software University](https://softuni.bg/trainings/2134/qa-automation-january-2019).

Please submit as homework a single zip / rar / 7z archive holding the source code or whatever approach is needed to finish your tasks. This exercise will be submitted together with next exercise after you receive it.

## Setup

For this exercises you need some additional software to start demo project. So before start doing this exercise follow this guide for setup and run demo project. Most of the steps need to be executed just once at the begging. All software **MUST** be installed with default settings (if you change anything in installation process it can result to not running demo)

1. Download and install [Postman desktop client](https://www.getpostman.com/downloads/)
2. Download and install [NodeJS](https://nodejs.org/en/) version v10.15.\*
3. Download and install [PostgreSQL](https://www.postgresql.org/download/) (add any password you want during the setup)
4. Download and extract demo zip file from SoftUni
5. In the root directory execute "npm install" command in the Command Line
6. Open file root/server/config/config.json and add PostgreSQL password for bot development and test
7. In the root directory execute "npm run db:create" command in the Command Line
8. In the root directory execute "npm run start:dev" for development environment

In the root directory execute "npm run start:test" for test environment

## End-points

After finishing setup, you have locally running API, which listening on port 3000. So you host URL should be <http://localhost:3000>. For every single request you need to add specific header with name "**G-Token**" and value "**ROM831ESV**", otherwise you will always receive response code 403 Forbidden.

* **GET endpoints –** always respond json format
  + **/books –** get all books in database
  + **/books/{id} –** get a single book with this id
  + **/books/search{{query}} –** return specific books, which fulfill search criteria
  + **/users –** get users in database
  + **/users/{id} –** get a single user with this id
* **POST endpoints** – All of them need to specify application/json for Content-Type
  + **/books –** you should provide title, author, isbn and publication date
  + **/users –** you should provide email, first and last name, household ID
  + **/households –** you should provide name
  + **/wishlists/{household}/book/{bookId} –** you should NOT add any body
* **DELETE endpoints –** need to be have basic authorization to perform this action
  + **/books/{id} –** delete a single book with provided id from database

# Postman

## Add first collection

Prepare collection of requests which need to:

1. **Add new household**
2. **Add two different user**
3. **Add two books for each user**
4. **Get all wishlist for created household**

Add at least an assert for each step. Add at least 3 different asserts for last step, which need to check that there is no duplicate book in the database.

# C#

Do same task in C# code

CreateHouseholds - 1

CreateUsers - 2

GetBooks - 3

AddBookToWishlist - 4

GetHouseholdBooks - 5

household.hasMany(models.user); >> Sofia

wishlist.belongsToMany(models.book, {through: 'wishlistBooks'}); >> Peter’s wishlist

user.belongsTo(models.wishlist);

book.belongsToMany(models.wishlist, {through: 'wishlistBooks'});